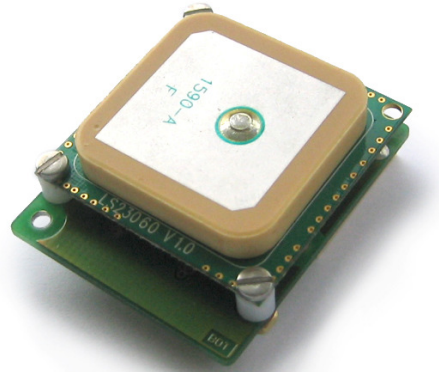




## *iSense GPS Module User Guide*

## *Document history*

| <i>Version</i> | <i>Date</i> | <i>Changes</i>  |
|----------------|-------------|-----------------|
| 1.0            | 12.05.2009  | Initial version |

## Contents

|        |                                                         |    |
|--------|---------------------------------------------------------|----|
| 1.     | About this User Guide .....                             | 4  |
| 2.     | General Description .....                               | 5  |
| 2.1.   | Handling and Security Instructions .....                | 6  |
| 2.2.   | Mounting the CPG Module on an iSense Module Stack ..... | 7  |
| 2.3.   | GPS Module Fix Indicator LED .....                      | 9  |
| 3.     | GpsModule API Description .....                         | 10 |
| 3.1.   | Constructor .....                                       | 10 |
| 3.2.   | Destructor .....                                        | 11 |
| 3.3.   | enable .....                                            | 11 |
| 3.4.   | disable .....                                           | 11 |
| 3.5.   | enabled .....                                           | 12 |
| 3.6.   | satellites .....                                        | 12 |
| 3.7.   | fix_state .....                                         | 13 |
| 3.8.   | PositionHandler interface .....                         | 13 |
| 4.     | GPS Module Demo Application .....                       | 15 |
| 4.1.   | Obtaining the GPS Module Demo Application .....         | 15 |
| 4.2.   | Compiling the GPS Module Demo Application .....         | 15 |
| 4.2.1. | Using Eclipse .....                                     | 15 |
| 4.2.2. | Using the Command Line .....                            | 18 |
| 4.3.   | GPS Module Demo Application functionality .....         | 19 |
| 4.4.   | Flashing the GPS Module Demo Application .....          | 23 |
| 5.     | References .....                                        | 26 |

## *1. About this User Guide*

In this user guide,

- files and folders are represented in the `Arial` typeface,
- code fragments, function names etc. are represented in the `Courier New` typeface,
- GUI elements such as button descriptions etc. are represented in “quotation marks”,
- titles of other documents are presented in *Italic* type.

This manual assumes that the reader has successfully installed the iSense development environment, and obtained the iSense standard firmware. For further information on these steps, consult the *Development Environment Setup User Guide* [1].

In addition, it is assumed that the user is familiar with the use of iShell. For further information on iShell, consult the *iShell User Guide* [2].

For further information on iSense firmware programming concepts and on application development, it is recommended to read the *Writing iSense Applications User Guide* [3].

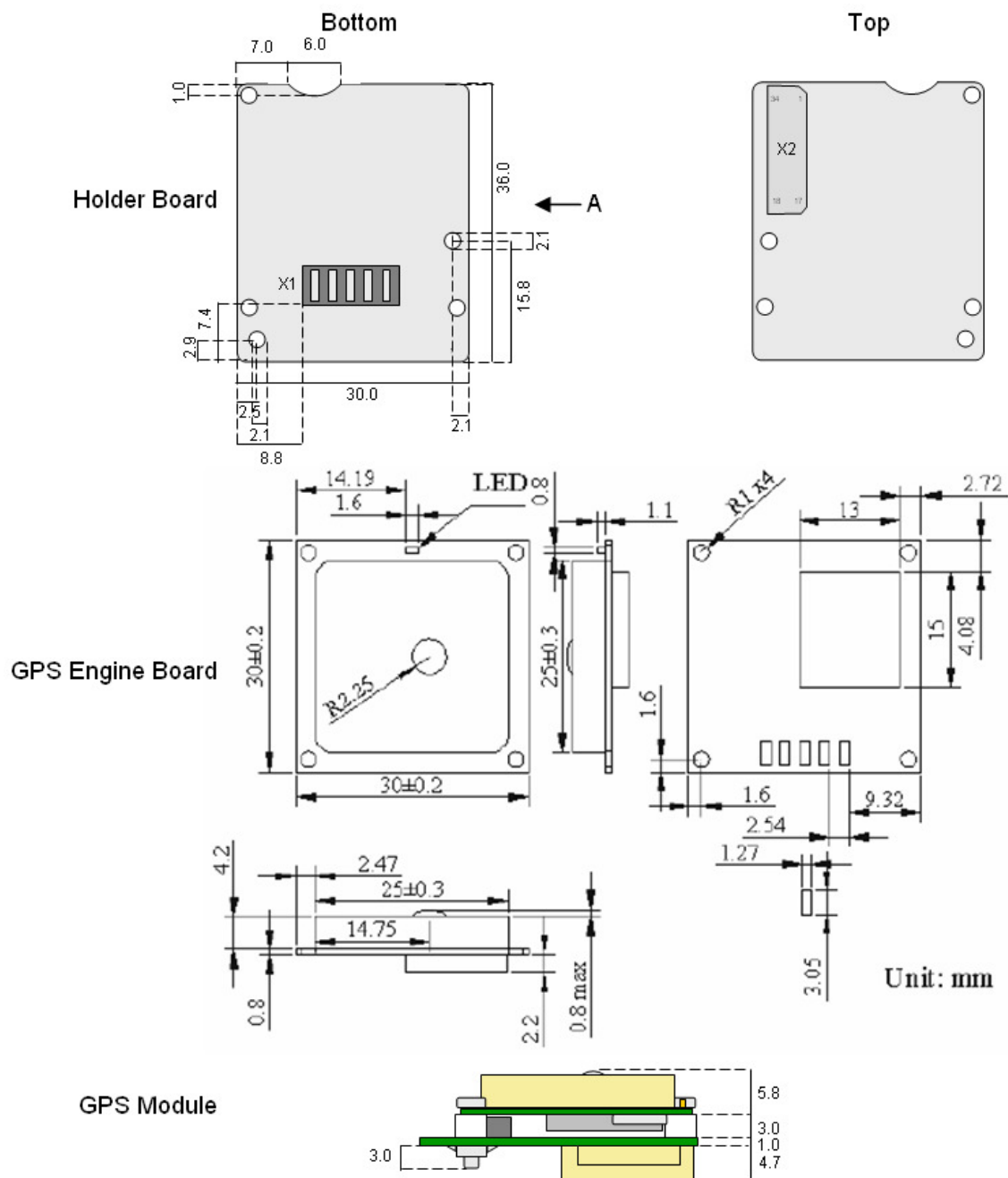
## 2. General Description

The iSense GPS Module supplies position information to iSense sensor nodes in all outdoor deployments. Like this, nodes equipped with the this module can not only serve as anchors for location protocols, but can also provide continuous location updates for mobiles sensor nodes

The iSense GPS Module is based on the SiRF Star 3 chipset. Due to its 200,000 correlators and 20 channel support it is known for its outstanding accuracy and sensitivity. It is capable of SBAS (WAAS, EGNOS, MSAS), and combines low power consumption with extremely fast fixing times.

The module features a high-performance patch antenna, a build-in micro battery to preserve system data for rapid satellite acquisition and an LED for GPS fix indication.

The GPS Module can be switched on and off by an iSense Core Module via an API call (see Section 3.3 and Section 3.4).



With the 34 pin connector (X2) on the top side, the GPS Module can be connected to other iSense modules such as the iSense Core Module.

### *2.1. Handling and Security Instructions*

Note that all iSense Modules are sensitive to electro-static discharge. Appropriate protection measures have to be taken.

If not protected by an appropriate housing, all iSense components must be protected from humidity, mechanical impact, and short circuiting by contact with conductive materials.

Note that all cable plugs and headers are designed to be extremely compact, and are not intended for frequent plugging and unplugging. Never pull the cables to remove the plugs from the corresponding modules, always pull the plug itself. If required, use an appropriate tool (or the pulling cord if available). Even though all plugs/headers are coded, be sure to pay attention to the proper orientation. Never apply force. Never use cables if their insulation is damaged.

Note that iSense modules are not intended for hot-plugging, (dis-)connecting modules from or to other modules in operation can result in a reset of the Core Module and other undesired effects.

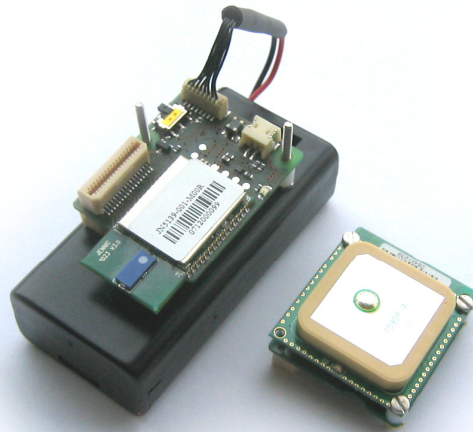
All delivered components are intended for use in research application. For using these components within other application domains, consult coalesenses prior to use.

Coalesenses products are not intended for use in life support systems, appliances or systems where malfunction of these products can reasonably be expected to result in personal injury, death or severe property or environmental damage. Coalesenses customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify coalesenses for any damages resulting from such use.

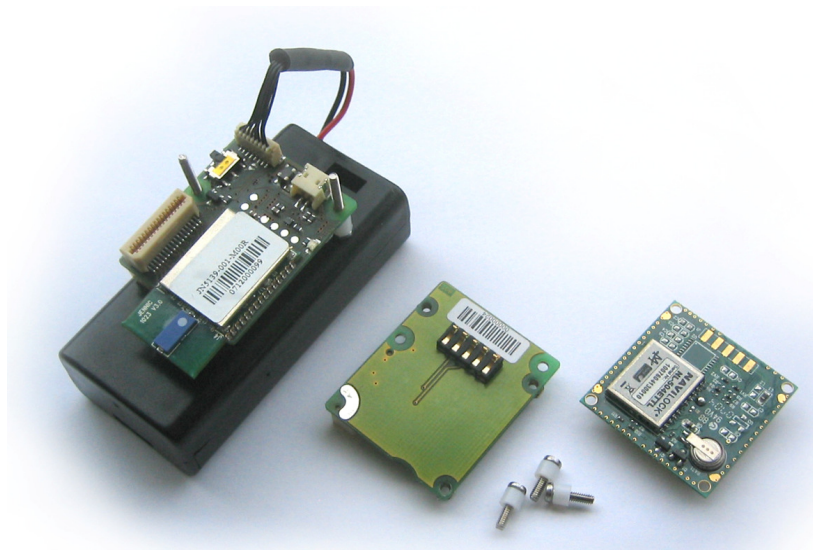
Do not connect components or devices to any iSense component that are not manufactured or distributed by coalesenses without prior consultation of coalesenses. Note that doing so regardless of consultation will void the warranty and the declaration of CE conformity. Coalesenses customers connecting third party devices do so at their own risk and agree to fully indemnify coalesenses for any damages or injury resulting from such use.

## *2.2. Mounting the CPG Module on an iSense Module Stack*

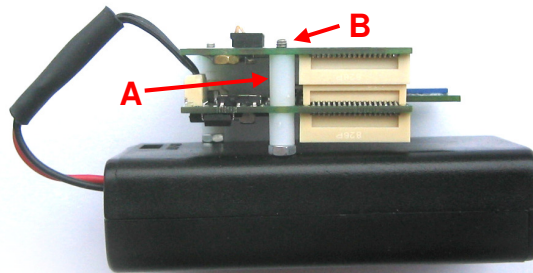
The iSense GPS Module consists of two parts, the GPS engine board and the holder board which it is mounted onto with three screws.



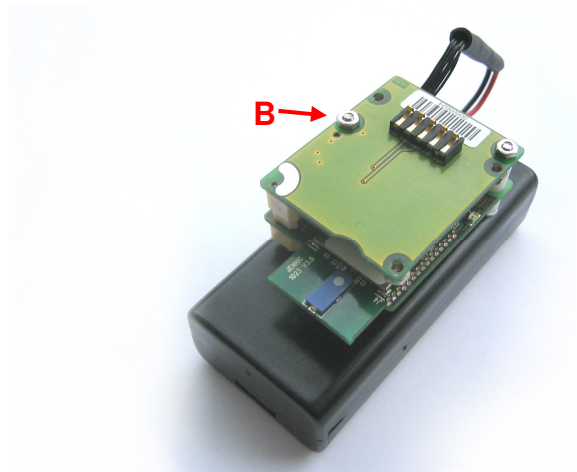
In order to mount the GPS Module onto a stack of other iSense Modules **and** to fix it with both stack mounting screws, you have to unmount the GPS engine board from the holder board. To do so, loosen the three screws that attach the GPS engine board to the holder board.



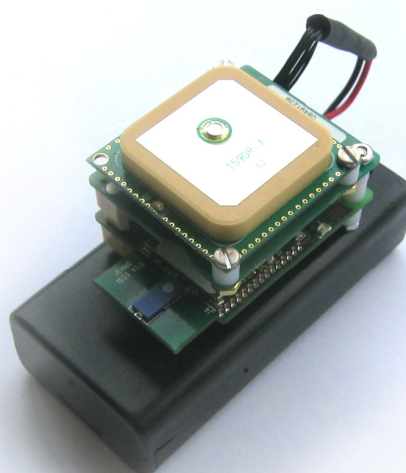
Use M2 x 8 mm spacers between the top iSense Module and the GPS holder board (A).



When mounting the iSense GPS holder module onto an iSense module stack, be sure that the stack mounting screw/bolt does not get into mechanical contact with the GPS engine board. To ensure this, make sure that the stack mounting screw/bolt does not stand out of the holder module more than 2mm (B) before you remount the GPS engine board onto the holder module. Screw together the iSense module stack.



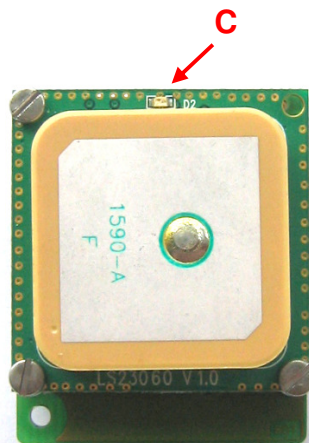
Finally, remount the GPS engine board onto the holder board.



### *2.3. GPS Module Fix Indicator LED*

The iSense GPS Module features a LED (C) that indicates the module's position fix state according to the table below.

| <i>LED activity</i>      | <i>GPS state</i>       |
|--------------------------|------------------------|
| off                      | module not active      |
| flashing fast            | no fix available       |
| flashing once per second | position fix available |



### 3. GpsModule API Description

The `GpsModule` class in the `isense` namespace contains all software functionality for operating the CPS Module-specific features, including

- switching the GPS Module on and off,
- registering for position fix information,
- reading out additional information such as fix state or satellite count.

The `GpsModule` class is defined in `src/isense/modules/gps_module/gps_module.h` in the `iSense` directory.

The iSense firmware is structured into a large number of software modules that can be (de-)activated separately. The below API description for each function specifies the required modules that must be activated to use the functions, and names both the web compilation module name as well as the according code define. An asterisk (\*) indicates that the respective module is activated in the iSense standard firmware.

The API description also indicates the make targets for which the different functions are available.

#### 3.1. Constructor

```
GpsModule::GpsModule(Os &os, PositionHandler* ph);
```

##### Description:

Initializes the `GpsModule` class. By default, the module is switched off. After the module is enabled, `ph.handle_position(...)` will be called upon position fix. For details on the `PositionHandler`, please refer to Section 3.8.

##### Parameters:

|                 |                                                                                                                   |
|-----------------|-------------------------------------------------------------------------------------------------------------------|
| <code>os</code> | Reference to the operating system class <code>Os</code>                                                           |
| <code>ph</code> | Reference the <code>PositionHandler</code> that shall be called upon a position fix. See Section 3.8 for details. |

##### Required modules:

|                  |                                               |
|------------------|-----------------------------------------------|
| GPS Module*      | <code>#define ISENSE_ENABLE_GPS_MODULE</code> |
| GPS Time Fixing* | <code>#define ISENSE_ENABLE_GPS_TIME</code>   |

##### Available for the targets:

JN5139R, JN5139R1

### *3.2.Destructor*

```
GpsModule::~GpsModule();
```

*Description:*

Disabled the GPS Module in case it was switched on.

*Parameters:*

none

*Required modules:*

GPS Module\*                    #define ISENSE\_ENABLE\_GPS\_MODULE

*Available for the targets:*

JN5139R, JN5139R1

### *3.3.enable*

```
bool GpsModule::enable();
```

*Description:*

Enables the GPS Module (if it is disabled). Enabling the GPS Module prevents the sensor node from going to sleep.

*Parameters:*

none

*Returns:*

true if the GPS Module was disabled before and is now enabled, false if the module was already enabled before

*Required modules:*

GPS Module\*                    #define ISENSE\_ENABLE\_GPS\_MODULE

*Available for the targets:*

JN5139R, JN5139R1

### *3.4.disable*

```
void GpsModule::disable();
```

*Description:*

Disables the GPS Module (if it is enabled).

*Parameters:*

none

*Required modules:*

GPS Module\*                    #define ISENSE\_ENABLE\_GPS\_MODULE

*Available for the targets:*

JN5139R, JN5139R1

### *3.5.enabled*

```
bool GpsModule::enabled();
```

*Description:*

Return the activity state of the GPS Module.

*Parameters:*

none

*Returns:*

true if the GPS Module is enabled, false otherwise

*Required modules:*

GPS Module\*                    #define ISENSE\_ENABLE\_GPS\_MODULE

*Available for the targets:*

JN5139R, JN5139R1

### *3.6.satellites*

```
bool GpsModule::satellites();
```

*Description:*

Returns the number of satellites used for the last position fix.

*Parameters:*

none

*Returns:*

Number of satellites, or ISENSE\_INVALID if the last fix was invalid or the GPS Module is disabled

*Required modules:*

GPS Module\*                    #define ISENSE\_ENABLE\_GPS\_MODULE

*Available for the targets:*

JN5139R, JN5139R1

### *3.7.fix\_state*

```
FixState GpsModule::fix_state();
```

*Description:*

Returns the fix state of the last position fix.

*Parameters:*

none

*Returns:*

ISENSE\_GPS\_FIX\_STATE\_INVALID

Fix not available or invalid, or the GPS Module is disabled

ISENSE\_GPS\_FIX\_STATE\_GPS

GPS SPS Mode, fix valid

ISENSE\_GPS\_FIX\_STATE\_DGPS

Differential GPS, SPS Mode, fix valid

ISENSE\_GPS\_FIX\_STATE\_DEAD\_RECKONING

Dead Reckoning Mode, fix valid

*Required modules:*

GPS Module\*                    #define ISENSE\_ENABLE\_GPS\_MODULE

*Available for the targets:*

JN5139R, JN5139R1

### *3.8.PositionHandler interface*

```
virtual void PositionHandler::handle_position(  
    uint8 latitude_degrees,  
    uint8 latitude_minutes,  
    uint16 latitude_frac,  
    DirectionIndicator lat_ind,  
    uint8 longitude_degrees,  
    uint8 longitude_minutes,  
    uint16 longitude_frac,  
    DirectionIndicator long_ind,  
    Time time  
    ) = 0;
```

*Description:*

In order to be called upon position fixes, a class must inherit from PositionHandler and has to overwrite the pure virtual handle\_position method.

After the GPS Module is enabled, `handle_position(...)` will be called upon position fix. The passed time will be the current UTC time if the “GPS Time Fixing” module is enabled in the firmware, or `INVALID_TIME` otherwise.

### *Parameters:*

|                                 |                                                                                                                                                                                                                  |
|---------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>latitude_degrees</code>   | latitude degrees (0...90)                                                                                                                                                                                        |
| <code>latitude_minutes</code>   | latitude minutes (0...59)                                                                                                                                                                                        |
| <code>latitude_fraction</code>  | fraction of latitude minutes (given in ten thousands)                                                                                                                                                            |
| <code>lat_ind</code>            | latitude indicator (northern or southern hemisphere):<br><br><code>ISENSE_GPS_LATITUDE_NORTH</code> northern hemisphere<br><br><code>ISENSE_GPS_LATITUDE_SOUTH</code> southern hemisphere                        |
| <code>longitude_degrees</code>  | longitude degrees (0...180)                                                                                                                                                                                      |
| <code>longitude_minutes</code>  | longitude minutes (0...59)                                                                                                                                                                                       |
| <code>longitude_fraction</code> | fraction of longitudes minutes (given in ten thousands)                                                                                                                                                          |
| <code>lat_ind</code>            | longitude indicator (western or eastern hemisphere):<br><br><code>ISENSE_GPS_LONGITUDE_WEST</code> western hemisphere<br><br><code>ISENSE_GPS_LONGITUDE_EAST</code> eastern hemisphere                           |
| <code>time</code>               | UTC time. Only works if the GPS time fixing functionality ( <code>ISENSE_ENABLE_GPS_TIME</code> ) is enabled. If so, a millisecond precision UTC time is passed. Otherwise, <code>INVALID_TIME</code> is passed. |

### *Required modules:*

|                  |                                               |
|------------------|-----------------------------------------------|
| GPS Module*      | <code>#define ISENSE_ENABLE_GPS_MODULE</code> |
| GPS Time Fixing* | <code>#define ISENSE_ENABLE_GPS_TIME</code>   |

### *Available for the targets:*

JN5139R, JN5139R1

## *4. GPS Module Demo Application*

The GPS Module demo application exemplifies how an application can make use of the `GpsModule` API. It shows how to

- register as a `PositionHandler`,
- switching the GPS Module on,
- deal with position fixes,
- query the number of satellites.

### *4.1. Obtaining the GPS Module Demo Application*

Go to coalesenses web site, click on “Downloads”, and then on “iSense Hardware”. In the “GPS Module” section, click on the icon at the right in the “GPS Module Demo Application” row, and download `GpsModuleDemoApplication.zip`.

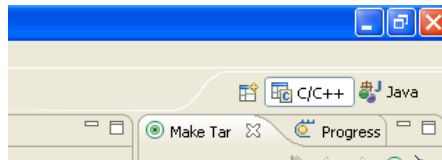
Extract the content of `GpsModuleDemoApplication.zip` to the `iApps` directory.

### *4.2. Compiling the GPS Module Demo Application*

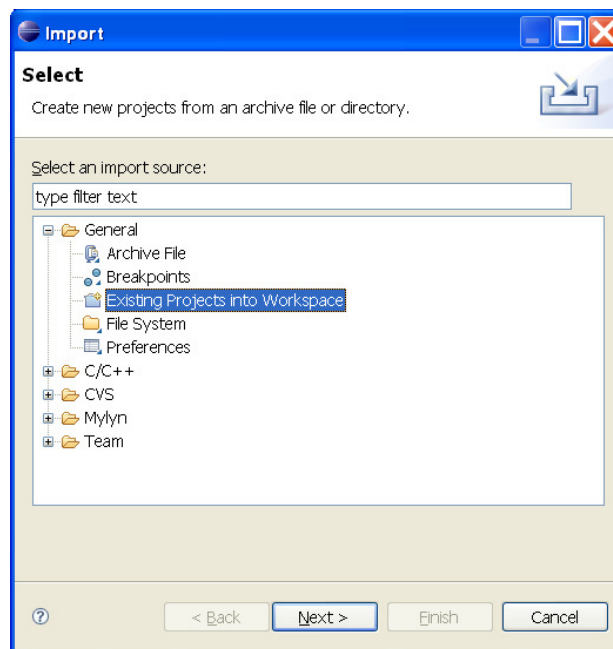
The next step is compiling the GPS Module Demo Application.

#### *4.2.1. Using Eclipse*

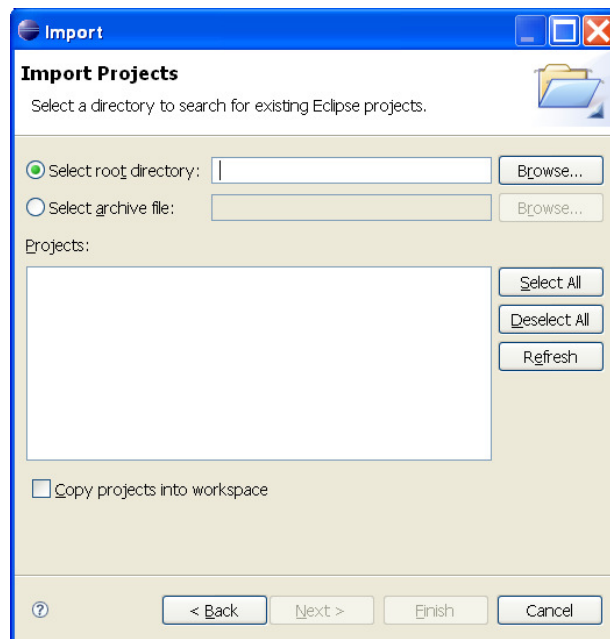
If you want to use Eclipse for compiling the GPS Module Demo Application, open Eclipse, and change to the “C++” perspective.



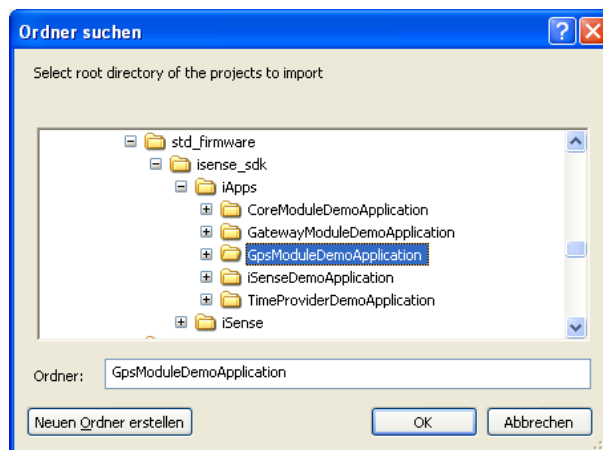
You can now import the application into Eclipse. Choose “File” → “Import” from the menu bar to open the “Import” dialog.



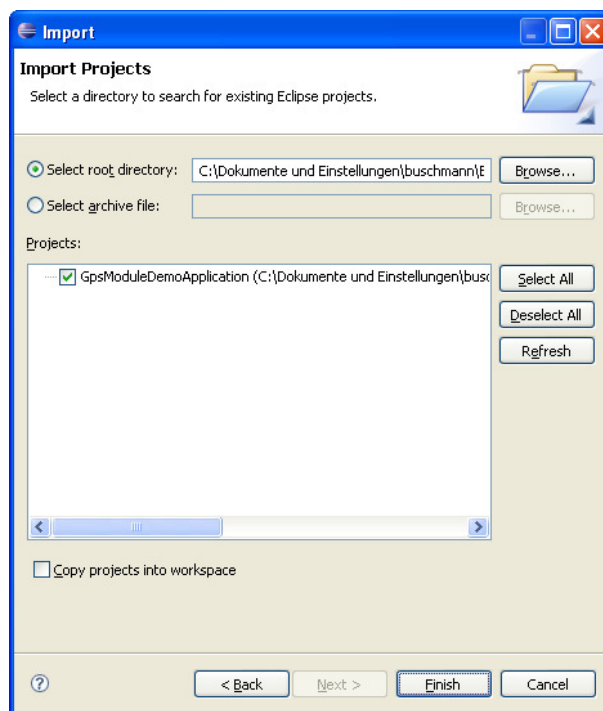
Choose “Existing projects into workspace”, and click on “Next”.



Click on the “Browse...”, and select the directory of the GPS Module demo application, i.e. GpsModuleDemoApplication in the iApps directory.



After doing so, the “GpsModuleDemoApplication” project should appear in the projects list of the “Import” dialog.

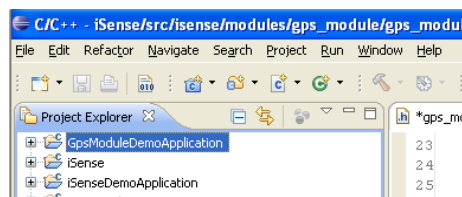


If it doesn't, the typical reason is that there is already a project called “GpsModuleDemoApplication” in Eclipse, i.e.

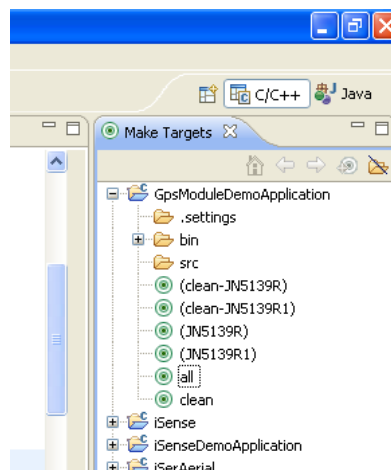
- you imported the GpsModuleDemoApplication before or
- you created or imported a project with the same name before.

Be sure not to check “Copy projects into workspace” in the above dialog. Finish the import by clicking on the “Finish” button.

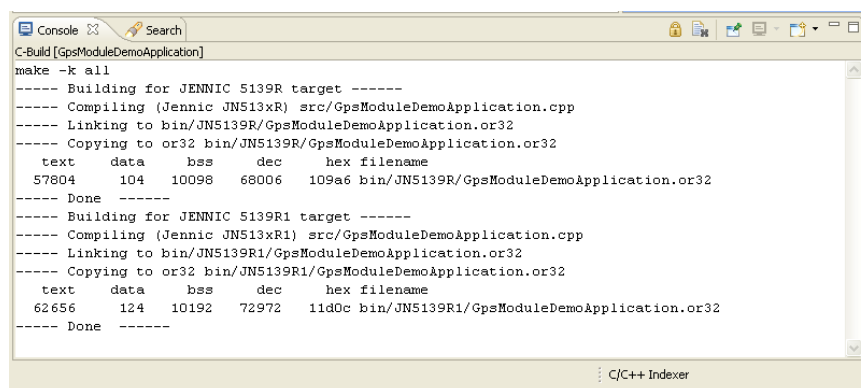
As a result, you will find the “GpsModuleDemoApplication” project in the “Project Explorer”.



In addition, the “GpsModuleDemoApplication” will appear in the “Make Targets” view. Double click on “all” to build the demo application for all targets.

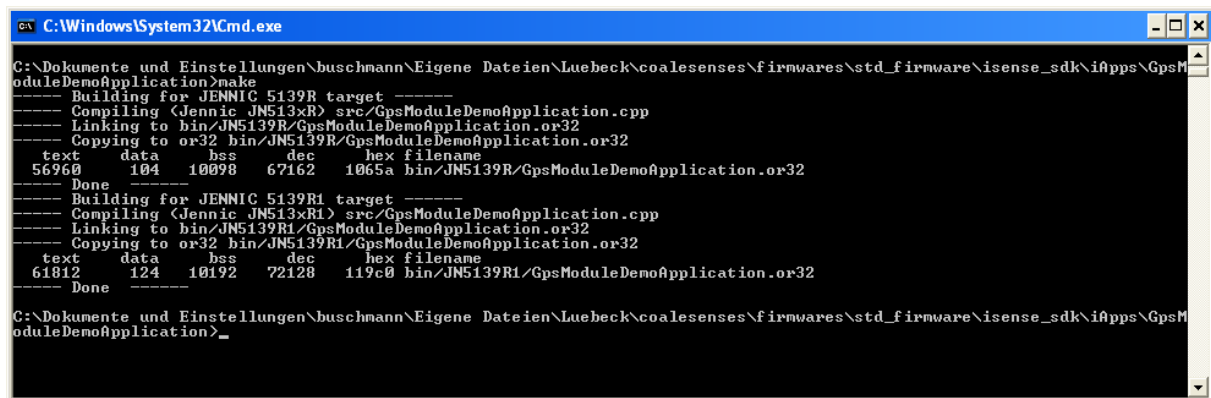


As a result, the “Console” view displays the compiler output during the build process. After that, it should look similar to the output depicted below.



### 4.2.2. Using the Command Line

If you want to use the command line tools for building the GPS Module Demo Application, open a console window, change to the `GpsModuleDemoApplication` directory in your `iApps` directory, and type “make”.



```
C:\Windows\System32\cmd.exe

C:\Dokumente und Einstellungen\buschmann\Eigene Dateien\Luebeck\coalesenses\firmwares\std_firmware\isense_sdk\ifapps\GpsModuleDemoApplication>make
----- Building for JENNIC 5139R target -----
Compiling (Jennic JN513xR) src/GpsModuleDemoApplication.cpp
Linking to bin/JN5139R/GpsModuleDemoApplication.or32
Copying to or32 bin/JN5139R/GpsModuleDemoApplication.or32
text  data  bss  dec  hex filename
56960  104  10098  67162  1065a bin/JN5139R/GpsModuleDemoApplication.or32
Done
----- Building for JENNIC 5139R1 target -----
Compiling (Jennic JN513xR1) src/GpsModuleDemoApplication.cpp
Linking to bin/JN5139R1/GpsModuleDemoApplication.or32
Copying to or32 bin/JN5139R1/GpsModuleDemoApplication.or32
text  data  bss  dec  hex filename
61812  124  10192  72128  119c0 bin/JN5139R1/GpsModuleDemoApplication.or32
Done
C:\Dokumente und Einstellungen\buschmann\Eigene Dateien\Luebeck\coalesenses\firmwares\std_firmware\isense_sdk\ifapps\GpsModuleDemoApplication>
```

After the build finished, the console output should look similar as shown above.

### 4.3. GPS Module Demo Application functionality

The source file of the GPS Module demo application is located at `GpsModuleDemoApplication/src/GpsModuleDemoApplication.cpp`. The class `GpsModuleDemoApplication` is derived from `isense::Application`, and additionally inherits from `isense::PositionHandler`.

```
class GpsModuleDemoApplication :
public Application,
public PositionHandler
{
public:
    GpsModuleDemoApplication(Os &os);
    // inherited from application, called upon device boot
    void boot();
    // inherited from isense::PositionHandler, called upon GPS Module
    position fix
    virtual void handle_position( uint8 latitude_degrees,
        uint8 latitude_minutes,
        uint16 latitude_frac,
        DirectionIndicator lat_ind,
        uint8 longitude_degrees,
        uint8 longitude_minutes,
        uint16 longitude_frac,
        DirectionIndicator long_ind,
        Time time );
private:
    GpsModule* gps_;
};
```

`GpsModuleDemoApplication` overwrites `isense::Application::boot()`, which is called upon the start up of the iSense sensor node.

```

void
GpsModuleDemoApplication::
boot ()
{
    //output a boot notification via a potentially connected Gateway Module
    os().debug("App::Booting GPS Demo Application");

    // allocate a GpsModule instance, pass this application as
    // PositionHandler
    gps_ = new GpsModule(os(), this);

    // if allocation was successful
    if (gps_!= NULL)
    {
        // enable GPS Module
        gps_>enable();

        // set logging mode to "Radio only"
        os().set_log_mode(ISENSE_LOG_MODE_RADIO);
    }
    else // output error message via a potentially connected Gateway Module
        os().fatal("Could not allocate GpsModule");
}

```

Within the `boot()` method, a `GpsModule` object is created, and a reference to the operating system and a pointer to the application are passed to the constructor. Like this, the application is registered at the `GpsModule` as the `PositionHandler`. This is possible because the application implements the pure virtual method from `isense::PositionHandler` (see Section 3.8).

If the allocation was successful, the GPS Module is enabled. This will start the position acquisition. Note that while the GPS Module is enabled, the sensor node is prevented from going to the sleep mode, because receiving position updates from the GPS Modules requires the node to be awake. As soon as the GPS Module is disabled again, the node can go to sleep again (unless other components prevent sleep, too).

After that, the log mode of the operating system is set the “Radio only” (no option but `ISENSE_LOG_MODE_RADIO` is set).

You can send character output to the outside world using the methods `Os::debug` and `Os::fatal`:

```

//-----
** Logs the given string depending on the log mode to a uart or radio.
*/
void Os::debug( const char *format, ...);

//-----
/** Logs the given string depending on the log mode to a uart or radio.
*/
void Os::fatal( const char *format, ...);

```

They work similar to the well-known `sprintf` functions in regular C, i.e. integer values etc. can be printed using the `%` notation. For example

```
uint16 i = 128;
os().debug("value of i is %d, hex=%x", i, i);
```

will output „value of i is 128, hex=0x80“. The destination of the debug output can be set using `Os::set_log_mode`, the corresponding constants can be found in `iSense/src/isense/os.h`.

```
/** Sets the log mode to the given value, e.g.
 * (ISENSE_LOG_MODE_UART0 | ISENSE_LOG_MODE_RADIO)
 */
void Os::set_log_mode( uint8 mode ) {log_mode_ = mode; }
```

By default, the output destination is set to UART0, i.e. the output is sent to a PC via a connected iSense Gateway Module.

In the `GpsModuleDemoApplication::boot()` method, the first boot notification message as well as the fatal message at the end are hence sent via a potentially connected iSense Gateway Module, while all other messages of the application will be sent out via the radio. These messages that are sent out via the radio can be picked up with an `iSerAerial` node (see Section 4.4).

After the `GpsModuleDemoApplication::boot()` method enabled the GPS Module and passed the itself as the `PositionHandler`, the GPS Module will start to acquire a valid position fix. Whenever a new fix is available, the `GpsModuleDemoApplication::handle_position(...)` method is called.

```
void
GpsModuleDemoApplication::
handle_location( uint8 latitude_degrees,
                uint8 latitude_minutes,
                uint16 latitude_frac,
                DirectionIndicator lat_ind,
                uint8 longitude_degrees,
                uint8 longitude_minutes,
                uint16 longitude_frac,
                DirectionIndicator long_ind,
                Time time )
{
    // convert DirectionIndicators into characters
    char indicators[4] ={'N', 'S', 'W', 'E'};

    // output via the radio
    os().debug("%c %d° %d.%d' %c %d° %d.%d' #sat=%d",
               indicators[lat_ind], latitude_degrees,
               latitude_minutes, latitude_frac,
               indicators[long_ind], longitude_degrees,
               longitude_minutes, longitude_frac, gps_->satellites());
```

```
// if the "GPS Time fixing" feature is enabled in the firmware
#ifdef ISENSE_ENABLE_GPS_TIME
    uint8 hour, min, sec, day, month;
    uint16 year, ms;
    // convert the time stamp into conventional date/time data
    time.to_date_time(year, month, day, hour, min, sec, ms);

    // output via the radio
    os().debug("UTC=%t (%d-%d-%d %d:%d:%d.%d)",
               &time, year, month, day, hour, min, sec, ms);
#endif
}
```

There, the received position is output using `Os::debug` and. For convenience, the `DirectionIndicator` is converted into characters by using the below values defined in `src/isense/modules/gps_module/gps_module.h` in the `iSense` directory.

```
typedef enum {
    ISENSE_GPS_LATITUDE_NORTH = 0,    /**< Northern hemisphere */
    ISENSE_GPS_LATITUDE_SOUTH = 1,    /**< Southern hemisphere */
    ISENSE_GPS_LONGITUDE_WEST = 2,     /**< Western hemisphere */
    ISENSE_GPS_LONGITUDE_EAST = 3      /**< Eastern hemisphere */
} DirectionIndicator;
```

In addition, the number of satellites involved in the last fix is queried from the GPS Module and output as well.

If the “GPS time fixing” feature is enabled in the firmware (`#ifdef ISENSE_ENABLE_GPS_TIME` will be true), the time stamp passed to `GpsModuleDemoApplication::handle_position(...)` is converted into conventional date and time information and is output via `Os::debug(...)`. Note that the date and time information is given with respect to UTC, i.e. it ignores local time zones and day light saving.

All in all, the `GpsModuleDemoApplication` enables the GPS Module and outputs the position via the radio upon an available position fix.

#### *4.4. Flashing the GPS Module Demo Application*

Running the GPS Module demo application involves two iSense sensor nodes, the “GPS node” and the “iSerAerial node”:

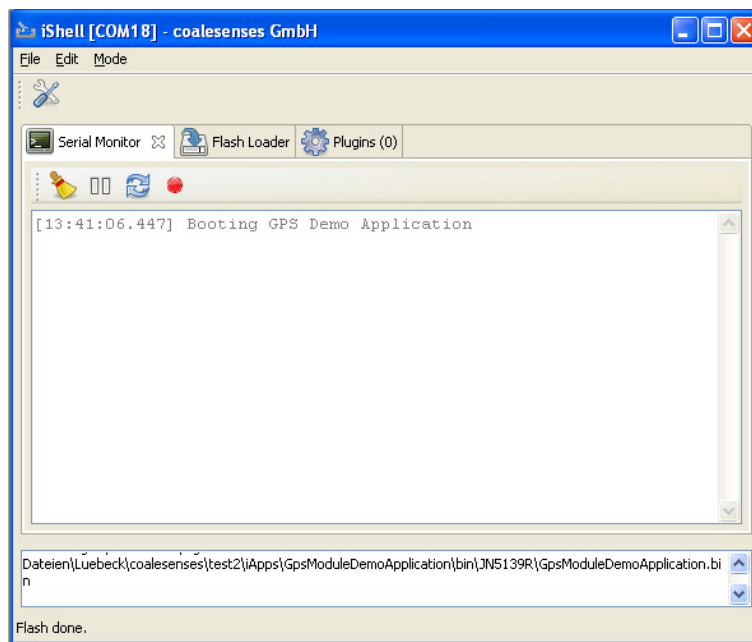
| <i>Node</i>     | <i>Configuration</i>                                                                                                                                                                                                                                                                                      |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GPS node        | <ul style="list-style-type: none"><li>• an iSense Core Module,</li><li>• an iSense Battery Module and</li><li>• an iSense GPS Module</li></ul> as well as <ul style="list-style-type: none"><li>• an additional iSense Gateway Module with USB or serial cable</li></ul>                                  |
| iSerAerial node | <ul style="list-style-type: none"><li>• an iSense Core Module and</li><li>• an iSense Gateway Module with a USB cable</li></ul> or <ul style="list-style-type: none"><li>• an iSense Core Module,</li><li>• an iSense Battery Module and</li><li>• an iSense Gateway Module with a serial cable</li></ul> |

First of all, disconnect the GPS Module from the GPS mode, attach the additional Gateway Module instead and plug the USB/serial cable to the PC. Be sure that the Core Module is switched on.

Start iShell, configure the correct communication port, and change to the “Flash Loader” view. Select the appropriate binary (depending on the chip version of the Core Module) according to the table below.

| <i>Chip</i> | <i>Binary file</i>                                                       |
|-------------|--------------------------------------------------------------------------|
| JN5139R     | iApps/GpsModuleDemoApplication/bin/JN5139R/GpsModuleDemoApplication.bin  |
| JN5139R1    | iApps/GpsModuleDemoApplication/bin/JN5139R1/GpsModuleDemoApplication.bin |

For details regarding the identification of the chip version, refer to Section 2.4 of the *iSense Core Module User Guide* [4]. Then flash the binary to the connected iSense sensor node. After flashing completed, the output line “Booting GPS Demo Application” should appear in the “Serial Monitor” view.

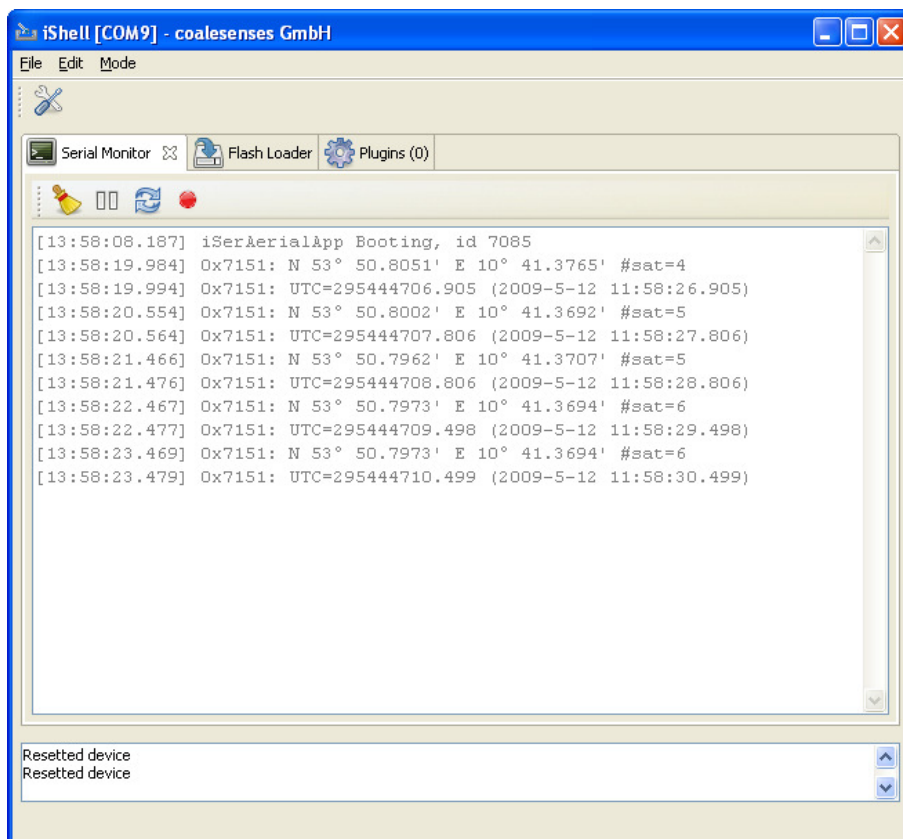


Close iShell, switch off the GPS node, disconnect the Gateway Module from the node, and connect the GPS Module instead.

Now, connect the iSerAerial node to the PC, switch it on, and restart iShell. Obtain the iSerAerial application binary [5] that matches the chip version of the Core Module of the iSerAerial node from the coalesenses website, and flash it to the iSerAerial node.

This makes the iSerAerial node to act as a Radio-to-Serial and Serial-to-Radio bridge. Besides other functions, it forwards debug or fatal message sent out via the radio to the connected iShell whenever it receives them.

Now, switch on the GPS node, place it in a location where it can receive GPS signals (e.g. at a window) and where wireless communication with the iSerAerial node is possible. The fix indication should flash fast for a while, and then change to flashing every second, i.e. whenever a position fix is available.



Observe output at the iShell to which the iSerAerial node is connected. At the time when the flashing mode changes, position information should appear in the “Serial Monitor” text field.

Position acquisition of the GPS node may take up to 90 seconds (usually less than 20 seconds). If the node does not acquire a fix after a certain time, try a different location.

If no output appears even though the GPS node flashes every second, try to move the iSerAerial node closer to the GPS node (both should be in the same room). If that does not help, be sure that both nodes operate on the same channel, i.e. that you did not set a different channel on any of the nodes.

## 5. References

- [1] coalesenses Development Environment Setup User Guide, online available at [http://www.coalesenses.com/download/UG\\_development\\_environment\\_setup\\_v1.9\\_web.pdf](http://www.coalesenses.com/download/UG_development_environment_setup_v1.9_web.pdf)
- [2] coalesenses iShell User Guide, online available at [http://www.coalesenses.com/download/UG\\_ishell\\_v1.3.pdf](http://www.coalesenses.com/download/UG_ishell_v1.3.pdf)
- [3] coalesenses Writing iSense Applications User Guide, online available at [http://www.coalesenses.com/download/UG\\_writing\\_isense\\_applications\\_v1.pdf](http://www.coalesenses.com/download/UG_writing_isense_applications_v1.pdf)
- [4] coalesenses iSense Core Module User Guide, online available at [http://www.coalesenses.com/download/UG\\_CM10X\\_1v0.pdf](http://www.coalesenses.com/download/UG_CM10X_1v0.pdf)
- [5] coalesenses iSerAerial application binary, online available at <http://www.coalesenses.com/index.php?page=isense-applications>

coalesenses GmbH  
Maria-Goeppert-Str. 1  
23562 Lübeck  
Germany

[www.coalesenses.com](http://www.coalesenses.com)  
[sales@coalesenses.com](mailto:sales@coalesenses.com)